3D Rasterization

Unifying Rasterization and Ray Casting

Simon Christoph Stein

Georg-August-Universität Göttingen

20 Dez 2011

Inhaltsverzeichnis

Teil I

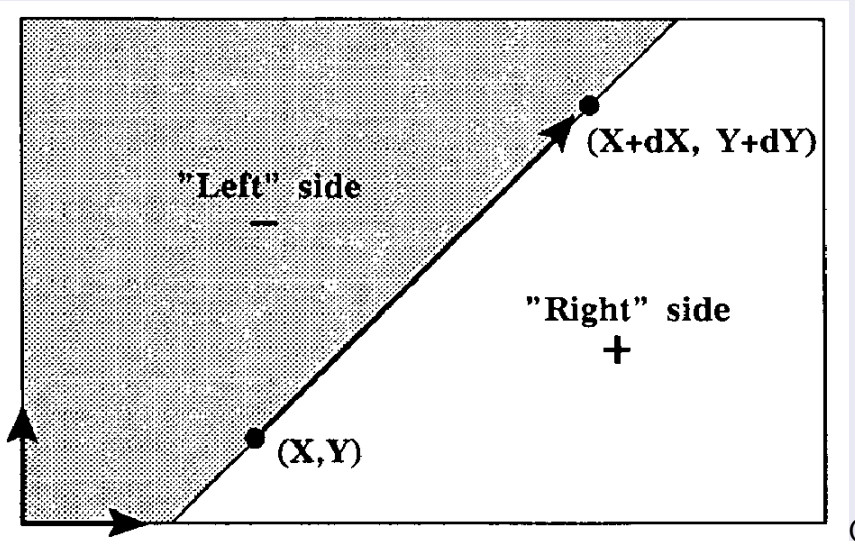
Traditionelle Renderverfahren

Rasterung

- Derzeitiger Standard für Echtzeitgrafik (Privatanwender)
- Renderingpipeline (Skizze):
  - Modeling & Viewing Transformation (optional: Belichtung)
  - Clipping & Projektion
  - (Hierachische) Rasterung & Early Z-Test, Interpolation von Vertex-Attributen, Pixel-Shading, Texturierung
  - Tiefentest, Stencil Operationen, Blending etc.
- **Rasterung:** Umwandlung von kontinuierlicher Beschreibung von Objekten im Bildraum (2D) in diskrete Rastergrafiken
- Traversierung der Szene:

```
Für jedes Dreieck{
    Für jedes Pixel{
    Wenn(Sichtbar(Pixel,Dreieck))
    Ausgabe
    }
}
```

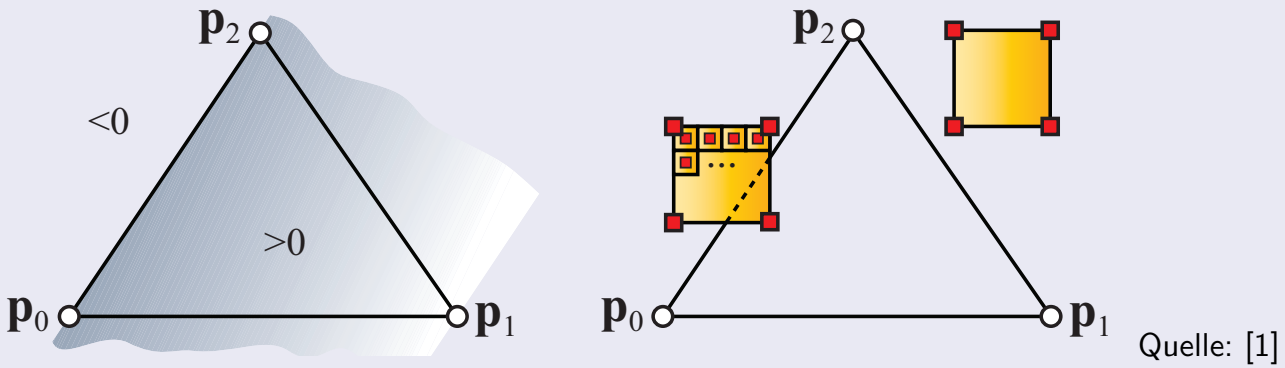
- Prüfung nötig, ob ein Pixel von einem gegebenen Dreieck überdeckt wird
- $\Rightarrow$ Effiziente Methode über 2D-Kantenfunktionen



Quelle: [4]

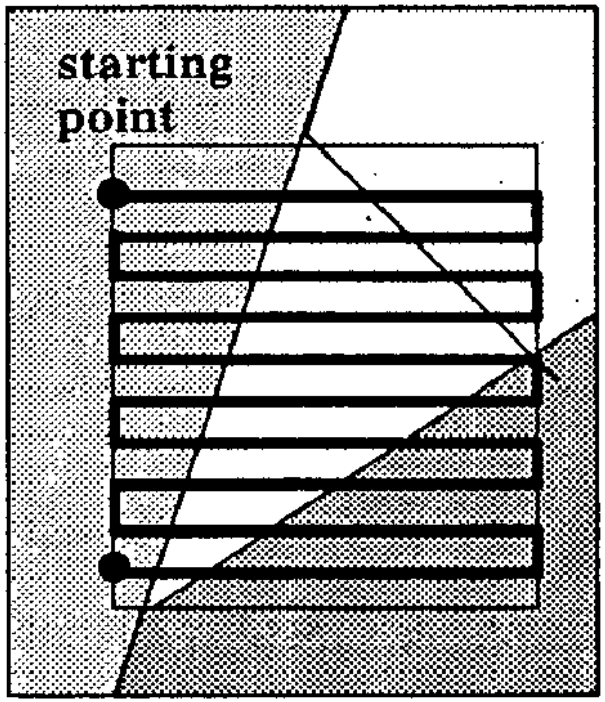
- $E(x,y) = (x-X)dY - (y-Y)dX = \left(\begin{pmatrix} x-X \\ y-Y \\ 0 \end{pmatrix} \times \begin{pmatrix} (X+dX)-X \\ (Y+dY)-Y \\ 0 \end{pmatrix} \right)_z$
  - $E(x,y) > 0$ wenn (x,y) auf der rechten Seite
  - $E(x,y) < 0$ wenn (x,y) auf der linken Seite
  - $E(x,y) = 0$ wenn (x,y) auf Linie

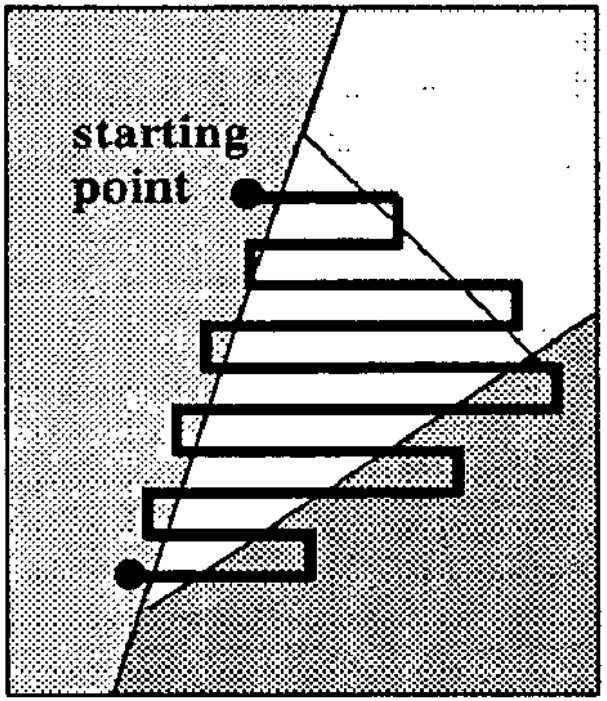
- Wenn $E_i(x, y) > 0$ für alle Kantenfunktionen $i = 0, 1, 2$ gilt
  $\Rightarrow$ Pixel liegt in Dreieck.
- Inkrementelle Berechnung möglich:
  - $E(x + 1, y) = E(x, y) + dY$
  - $E(x, y + 1) = E(x, y) - dX$



Quelle: [1]

- Wichtig für eine hohe Geschwindigkeit ist **Binning**:
  Schnelle Schätzung, welche Pixelbereiche für ein gegebenes Dreieck geprüft werden müssen.
- Üblich ist die Prüfung an den vier Eckpunkten eines quadratischen „Bins“ gegen ein Dreieck





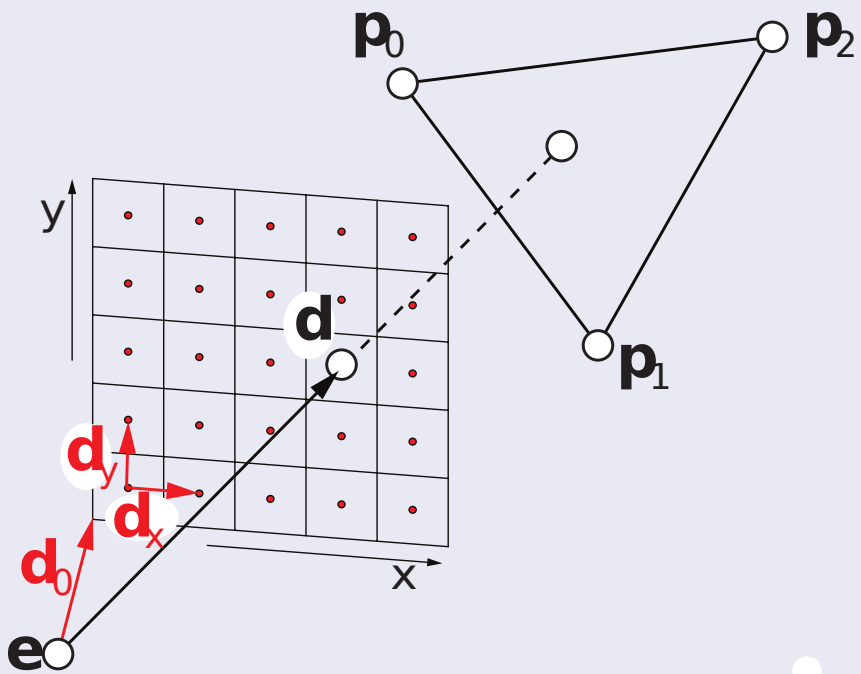
Quelle: [4]

- Naiver Binning-Algorithmus: Bounding Box (linke Abb.)
- Besser über hierarchische Strukturen (z.B: Quad Trees)

Raytracing

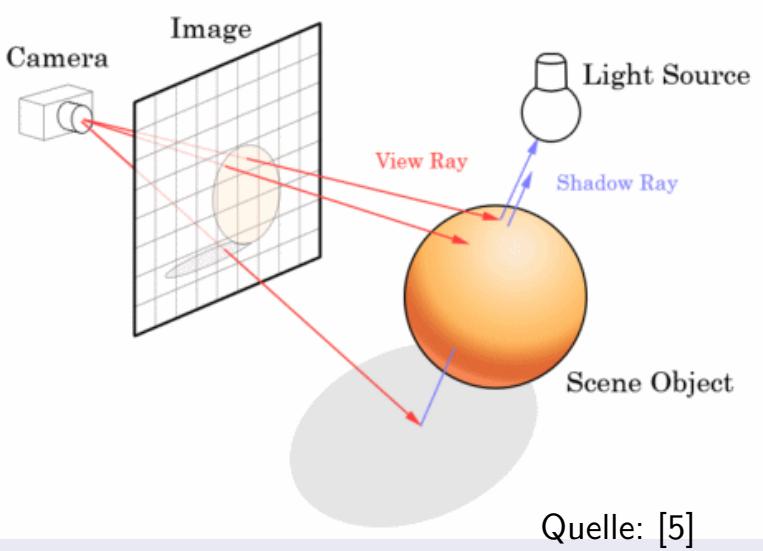
- Aktuell bevorzugt im professionellen Bereich (Kinofilme, CAD)
- Sichtbarkeitstests durch Strahlverfolgung in Welt- oder Kamerakoordinaten
- Für jeden Pixel wird ein Strahl vom Auge des Betrachters in die Szene geschickt
- Traversierung der Szene:

```
Für jedes Pixel{
    Für jedes Dreieck{
    Wenn(Sichtbar(Pixel,Dreieck))
    Ausgabe
    }
}
```



Quelle: [1]

- Um nicht für jeden Strahl gegen jedes Dreieck testen zu müssen werden optimierte Raumstrukturen verwendet, z.B. Octree, kd-tree, BVH oder 3D Gitter
- Rechenaufwand steigt logarithmisch mit Anzahl der Primitive (Rasterung: linear) und linear in Anzahl der Pixel.



Quelle: [5]

- Wird nur die Sichtbarkeitsprüfung über die Primärstrahlen verwendet, spricht man oft von **Ray Casting**
- Wenn vom Auftreffpunkt weitere Sekundärstrahlen ausgesandt werden (z.B. reflektierter und gebrochener Strahl, Schattenstrahl) verwendet man den Begriff **Raytracing**
- $\Rightarrow$ Gutes Framework für fotorealistische Beleuchtung

Teil II

3D Rasterung

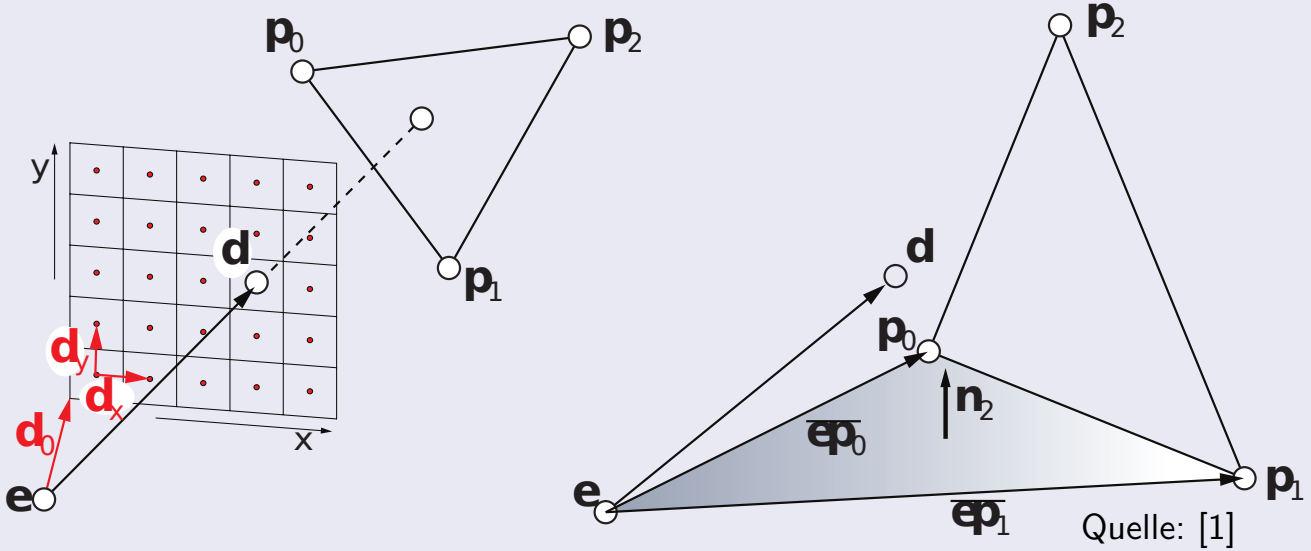
3D Rasterung

3D Rasterung

- **Ziel:** Algorithmus, der Eigenschaften sowohl von Raytracing als auch Rasterung besitzt und Konzepte zwischen beiden Verfahren austauschen kann.
- **Idee:** Test, ob Pixel von 2D Dreieck bedeckt wird ist äquivalent dazu, ob ein Strahl vom Beobachter durch diesen Pixel das Dreieck schneidet.
  $\Rightarrow$ Verwendung einer Ray Casting Variante für 3D Rasterung
- 3D Rasterung arbeitet direkt in Welt- oder Kamerakoordinaten und verwendet daher durchgängig Fließkommaoperationen
- $\Rightarrow$ Für Verwendung auf hochparallelisierten Architekturen wie Intel's *Knights Ferry* (vormals *Larrabee*) gedacht.

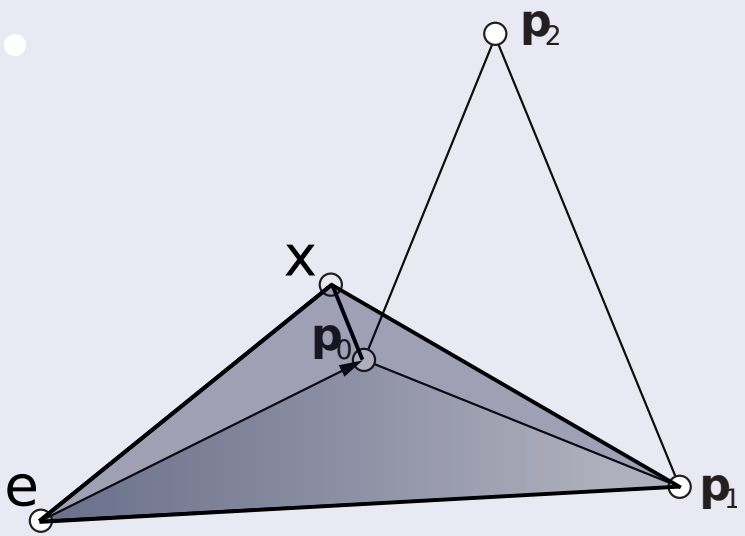
3D Kantenfunktion

- Wir wollen eine 3D-Kantenfunktion definieren.



Quelle: [1]

- Annahme: Beobachter im Ursprung $\vec{e} = 0$
- Strahlrichtung $\vec{d} = \vec{d}_0 + x\vec{d}_x + y\vec{d}_y$
- Dreieck mit Vertices $\vec{p}_0, \vec{p}_1, \vec{p}_2$
- Normalenvektor $\vec{n}_2 = \vec{p}_1 \times \vec{p}_0$



- Spatprodukt $V_2(\vec{x}) = n_2 \cdot \vec{x} = (\vec{p}_1 \times \vec{p}_0) \cdot \vec{x}$
  liefert Volumen des Tetraeders, der von $\vec{p}_0, \vec{p}_1, \vec{x}$ und $\vec{e}$ aufgespannt wird. (Skalenfaktor $\frac{1}{6}$ unwichtig)
- $V_2(\vec{x}) > 0$ für alle $\vec{x}$ im Halbraum der $\vec{p}_2$ enthält.

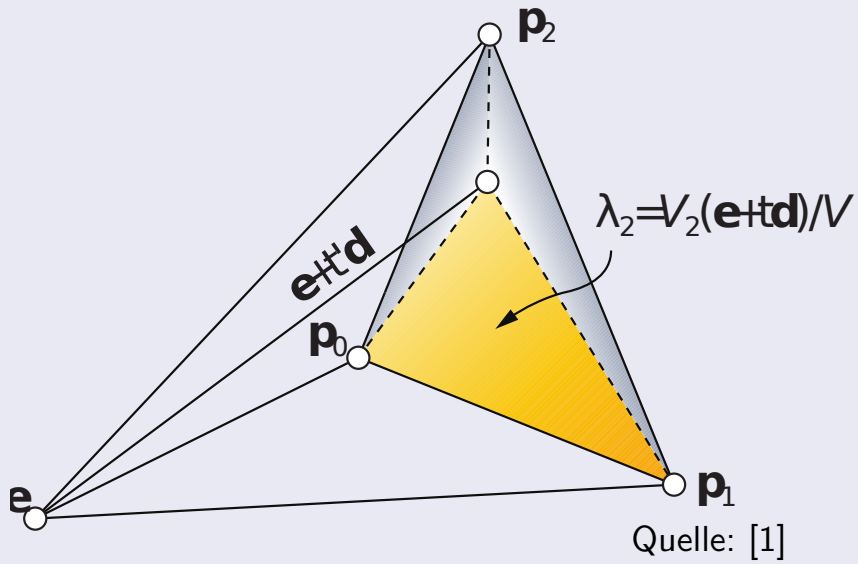
Definitionen:

- $\vec{n}_i := \vec{p}_{(i+2) \mod 3} \times \vec{p}_{(i+1) \mod 3}$
- $V_i(\vec{x}) := \vec{n}_i \cdot \vec{x}$ mit $i = 0, 1, 2$
- $V := \vec{p}_j \cdot \vec{n}_j$ für $j = 0, 1, 2$ beliebig
  Gesamtvolumen des Tetraders, der vom Dreieck mit dem Ursprung aufgespannt wird

- Der Strahl $\vec{e} + t'\vec{d}, t' > 0$ trifft das Dreieck, wenn alle vier Volumina das gleiche Vorzeichen haben.
- $V_i, V > 0$, Strahl trifft auf Frontfläche des Dreiecks
- $V_i, V < 0$, Strahl trifft auf Rückseite des Dreiecks

Am Schnittpunkt gilt:

- $V = V_0(t\vec{d}) + V_1(t\vec{d}) + V_2(t\vec{d})$
- $t = V/(V_0(\vec{d}) + V_1(\vec{d}) + V_2(\vec{d}))$



Quelle: [1]

- Baryzentrische Koordinaten des Schnittpunkts leicht berechenbar:
- $t\vec{d} = \lambda_0\vec{p}_0 + \lambda_1\vec{p}_1 + \lambda_2\vec{p}_2$
  mit $\lambda_i = V_i(\vec{d})/(V_0(\vec{d}) + V_1(\vec{d}) + V_2(\vec{d}))$

- Strahlparameter t nicht für Schnitttest oder baryzentrische Koordinaten benötigt..
- .. aber möglich, über t den Schnittpunkt in Welt-/ Kamerakoord. direkt zu berechnen und einen Tiefentest durchzuführen (ähnlich Z-Buffer).

- Definition der Ableitungen $V_{i,x}, V_{i,y}$:
- $V_i(\vec{d}) = \vec{n}_i \cdot \vec{d} = \vec{n}_i \cdot (\vec{d}_0 + x\vec{d}_x + y\vec{d}_y)$

  $V_{i,x} = \frac{\partial V_i(\vec{d})}{\partial x} = \vec{n}_i \cdot \vec{d}_x, V_{i,y} = \frac{\partial V_i(\vec{d})}{\partial y} = \vec{n}_i \cdot \vec{d}_y$
- Effiziente Berechnung im Fall eines planaren Viewports möglich, da $V_i(\vec{d}_0) + xV_{i,x} + yV_{i,y}$ für ein gegebenes Pixel (x, y). (Auch für komplexere Viewports lässt sich oft ein inkrementelles Schema finden).

Vergleich 2D/3D Rasterung

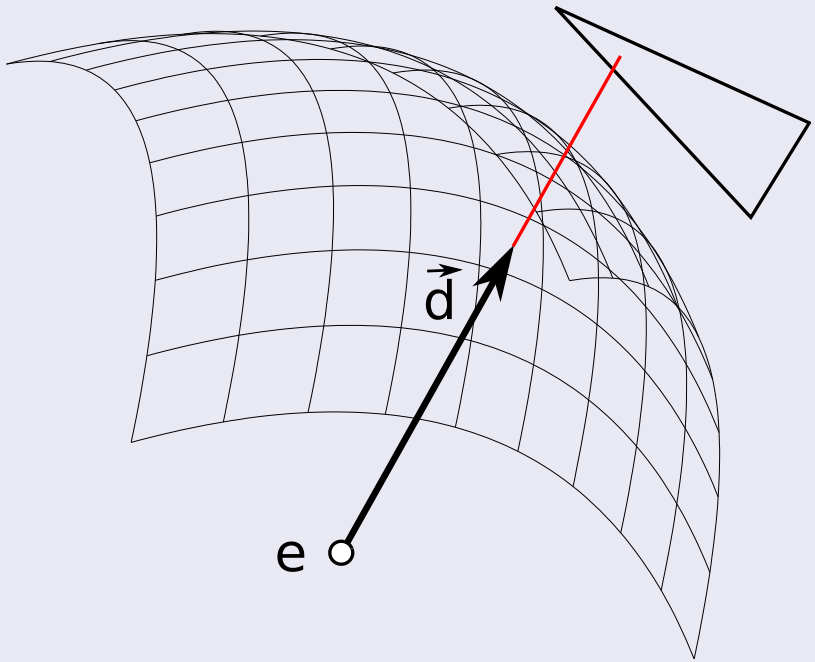
- Beide nutzen Kantenfunktionen zur Sichtbarkeitsprüfung
- 3D Rasterung kann schneller sein, insbesondere bei Berechnungen von perspektivisch korrekten Baryzentrischen Koordinaten.
- 3D Rasterung erfordert keine Projektion und kein Clipping.
  $\Rightarrow$ Auch nicht-planare Viewports möglich (entspricht Verzerrungen / nichtlinearer Projektion)
- Rechteck für „Binning“ in 2D, Pyramidenstumpf in 3D
- 2D Rasterung arbeitet mit Fixed-Point Arithmetik, während 3D Rasterung Fließkomma-Arithmetik benötigt.

Teil III

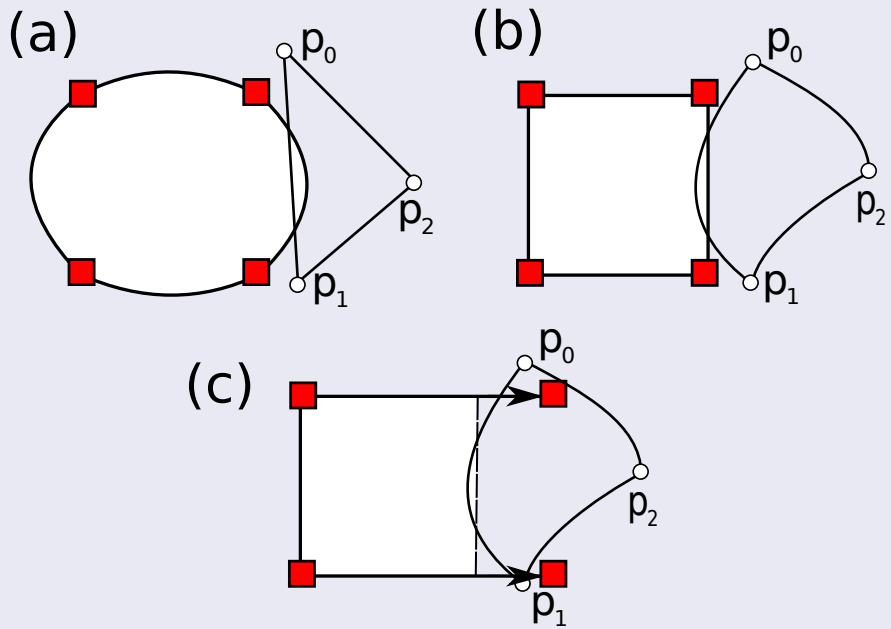
Anwendung von 3D Rasterung

Nicht-planare Viewports

- Linsenverzerrungen einfach möglich, indem „Sichtfenster“ verzerrt wird.
- Beispiel: Weitwinkeleffekt mit Paraboloid $f(x, y) = \frac{1}{2} - \frac{1}{2} \cdot (x^2 + y^2)$
- Fenstervektor $\vec{d} = (x, y, f(x, y))^T$



- Aber zu beachten: Binning muss geändert werden, da nur an Ecken eines Bins geprüft wird, dieser aber verzerrt wurde.



- Lösung: Virtuelle Verschiebung der für die Prüfung eines Dreiecks relevanten Ecken eines Bins.
- Nötige Verschiebung ergibt sich aus der Krümmung des Paraboloiden.
- Ähnliche Strategien können auch für andere nichtlineare Projektionen gefunden werden.

Antialiasing und eindeutiges Rendern

Kantenglättung

- Kantenglättung wie bei 2D Rasterung möglich

  $\rightarrow$ Im Paper Verwendung von Multi-Sampling (MSAA) und Coverage-Sample-Anti-Aliasing (CSAA)

Eindeutiges Rendern

- Bei 2D Rasterung existieren Konsistenzregeln (*top-left-convention*) um ein Pixel das von mehreren Dreiecken geschnitten wird eindeutig zuzuordnen.

  $\rightarrow$ Jedes Pixel wird genau einmal einem Dreieck zugeordnet
  $\rightarrow$ Verhindert Bildfehler bei Blending / Transparenzen
- Konsistenzregeln auf Basis der Ableitungen $V_{i,x}$, $V_{i,y}$ auch bei 3D Rasterung möglich
- Für Raytracing ist bisher kein vollständig konsistenter Renderalgorithmus bekannt.

Kontinuum zwischen Ray Casting und Rasterung

- Rasterung: Bearbeite alle Dreiecke der Szene in beliebiger Reihenfolge
- Raytracing: Für jeden Strahl nur die minimal nötige Anzahl von Dreiecken auf dem Weg bearbeiten.

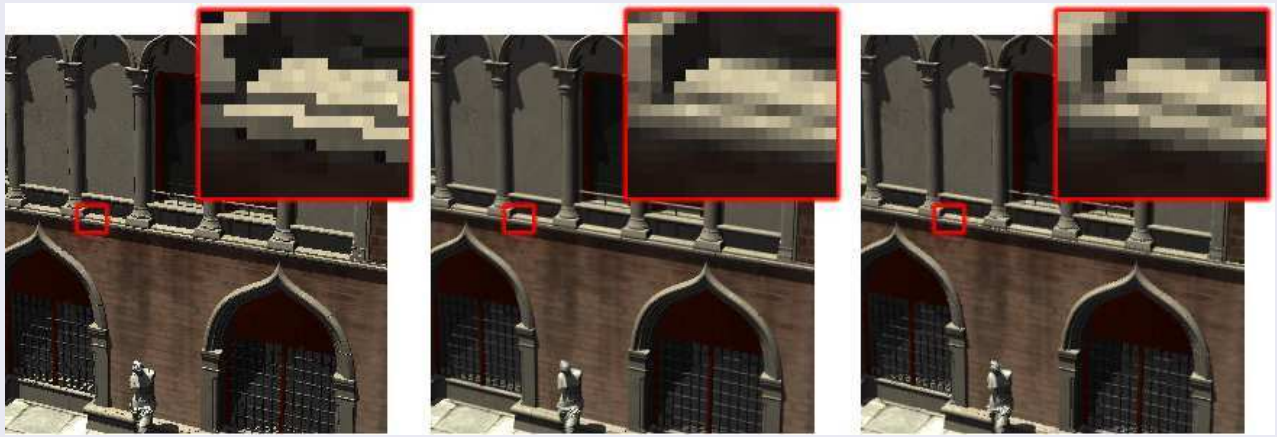
- Mittelweg: Bearbeite jeweils alle Dreiecke in einem Pyramidenstumpf
- Pyramidenstumpf gesamter Bildschirm: Reine Rasterung
- Algorithmus unterstützt aber auch Frustum- und Occlusion Culling unter Nutzung einer BVH, sowie Strahl-Dreieck-Schnittpunktstests auf Pixelebene
- Außerdem rekursive Unterteilung eines Pyramidenstumpfs möglich (mit folgender 3D-Rasterung für Dreiecke innerhalb des Stumpfes oder Per-Pixel-Tests)
- Alle Renderoptionen mit und ohne Binning möglich

Bilder

Nichtlineare Verzerrungen



Kantenglättung



Quelle: [1]

Fazit

- 3D Rasterung als Rendermethode im Kontinuum zwischen 2D Rasterung und Ray Casting.
- Fließkommaoperationen statt Fixed-Point-Arithmetik
- Mit 3D Rasterung lassen sich Konzepte zwischen Ray Casting und Rasterung erfolgreich austauschen
- Kann merklich schneller sein als herkömmliche 2D Rasterung
- Möglicher Vorteil in der Zukunft: Berechnung von Globaler Beleuchtung (bzw. globalen Effekten)
  $\rightarrow$ In 2D-Rasterung sehr schwer durchzuführen. In herkömmlichem Raytracing hoher Rechenaufwand.
- Die Autoren des Papers hoffen die bisherige Rendering-Pipeline zu vereinfachen.

Teil IV

Quellen

Quellen

1. C. Dachsbacher, P. Slusallek, T. Davidovic, T. Engelhardt, M. Philips and I. Georgiev. *3D Rasterization - Unifying Rasterization and Ray Casting*. Technical report, Universität Stuttgart, 2009
2. A. Appel. *Some techniques for shading machine renderings of solids.* AFIPS '68 (Spring): Proceedings of the April 30-May 2, 1968, spring joint computer conference
3. J. Pineda. *A parallel algorithm for polygon rasterization.* Computer Graphics (Proceedings of SIGGRAPH '88), 22(4), 1988
4. T. Whitted. *An improved illumination model for shaded display*. Communication of the ACM, 23(6), 1980
5. *Real Time Raytracing*, Jeff Atwood, 10. März 2008, `http://www.codinghorror.com/blog/2008/03/real-time-raytracing.html`